

## **REMARKS**

Claims 1-20 remain pending in the present application as amended. Independent claims 1, 8, and 14-20 have been amended. No claims have been added or canceled. No new matter has been added. Claims amendments regarding the unmanaged code, the managed code, and the common language run-time environment of a framework operating on the computing device is supported in the application as filed at least at paragraph 0008, paragraph 0009, and paragraph 0057.

### **Telephone Conversation With Examiner**

Examiner Joseph is thanked for the telephone conversation conducted on July 30, 2008. Proposed claim amendments were discussed. The cited art was discussed. The concepts of managed and unmanaged code were discussed. The recitation of computer-readable medium in claims 14-20 was discussed. No agreements were reached.

### **Amendments To Claims 14-20**

During the aforementioned telephone conversation, Examiner noticed that claims 14-20 may be subject to a 35 U.S.C. § 101 rejection. In anticipation of such a rejection, without prejudice or disclaimer, claims 14-20 are amended to recite a “computer-readable storage medium.”

### **Claim Objections And Rejections**

The Examiner has objected to claim 8 for the reason that a semi-colon is missing. Applicants have added the semi-colon to claim 8.

The Examiner has rejected claims 1, 2, 4-9, 11-15 and 17-20<sup>1</sup> under 35 USC § 102 as being anticipated by Morita et al. (U.S. Patent No. 5,959,712). In addition, the Examiner has

---

<sup>1</sup> The examiner states the rejected claims as being 1, 2, 4-10, 11-15 and 17-20, although claim 10 is also rejected under 35 USC § 103. Accordingly, Applicants will presume that claim 10 is rejected under section 103 and not 102.

rejected dependent claims 3, 10, and 16 under 35 USC § 103 as being obvious over the Morita reference.

Applicants respectfully traverse the Section 102 and 103 rejections. In particular, Applicants respectfully submit that the Morita reference fails to disclose or suggest transferring a pointer to shared memory between an unmanaged component and managed code of an application in the manner recited in amended independent claims 1, 8, and 14.

In the present application, pen data from a digitizer of a computing device or information derived from such pen data is employed by an application on the computing device with managed code. As should be understood, the pen data relates to movement of a stylus with respect to the digitizer, and thus includes information relating to one or more locations of the stylus with respect to the digitizer. Although the application employs managed code to process the pen data, the computing device employs a component (such as a DLL component) to initially receive the incoming pen data, where the component is not managed code but instead is written as unmanaged code.

As should be appreciated by the relevant public, managed code and unmanaged code relate to use of a framework on the computing device such as the .NET framework supplied by MICROSOFT Corporation of Redmond, Washington. Such .NET framework in particular includes a large library of pre-coded solutions to common programming problems, a runtime or virtual machine that manages the execution of programs written specifically for the framework, and a set of tools for configuring and building applications. The pre-coded solutions that form the framework's Base Class Library cover a large range of programming needs in a number of areas, including user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. The class library is used by programmers who combine it with their own code to produce applications.

Programs written for the .NET Framework (i.e., 'managed code') execute in a software environment that manages the program's runtime requirements. Also part of the .NET

Framework, this runtime environment is known as the Common Language Runtime (CLR). The CLR provides the appearance of an application virtual machine so that programmers need not consider the capabilities of the specific processor that will execute the program. As a result, managed code is executed under the CLR in a manner independent of the processor of the computing device. In contrast, 'unmanaged code' is not written for such framework and is not executed under the CLR. Instead, such unmanaged code is native to and executed directly by the processor of the computing device. Put another way, unmanaged code is written for a specific type of processor, while managed code is not likewise written for a specific type of processor.

Because interaction between managed and unmanaged code is commonly required, such as for example to transfer pen data from a digitizer to a .NET application, a framework such as the .NET Framework must provide mechanisms to access functionality that is implemented in programs that execute outside the .NET environment. For example, and as set forth in the application at paragraph 0009, a callable run-time wrapper may be invoked to retrieve information from a native component, where every call from managed space to the native component passes through standard interop data marshalling (data transfer) as established by the runtime callable wrapper. However, such standard interop data marshalling (data transfer) as employed in the .NET framework, for example, is relatively slow and therefore can be an impediment to high data throughput.

Accordingly, in various embodiments of the present innovation, data transfer is instead achieved by way of shared memory and pointers. Significantly, by doing so, packet transfer speed for pen data information may be increased by a factor of 2-5 times as compared with the standard CLR interop scheme (paragraph 0055).

As now recited in claim 1, the present innovation may be embodied as a process for transferring pen data between unmanaged and managed code on a computing device. The unmanaged code is code native to and executed directly by a processor of the computing device, and the managed code is code executed in a common language run-time environment of a

framework operating on the computing device. Thus, the common language run-time environment of the framework executes the managed code independent of a type of the processor of the computing device.

In the process of claim 1, pen data is received in a component on the computing device, where the component is written in unmanaged code. The pen data is generated by a digitizer of the computing device upon movement of a stylus with respect to a surface of the digitizer, and the pen data includes at least one location on the digitizer of the stylus. Information related to the pen data is transferred to a shared memory on the computing device designated to be shared between unmanaged code and managed code, and the component written in unmanaged code transfers a pointer that points to the information in the shared memory to an application on the computing device written at least in part in managed code. Thus, the application written at least in part in managed code retrieves the information from the shared memory by way of the transferred pointer.

Independent claims 8 and 14 each recite subject matter akin to the process of claim 1, albeit in the form of a system (claim 8) and a computer-readable medium (claim 14).

The Morita reference deals with systems and methods for employing pen data from a digitizer. In the Morita reference, a coordinate detecting section 21 detects a coordinate value for a position indicated by a coordinate indicator 3 and a switch status. A function selecting means 22 determines that a menu provided on a tablet 2 is read from the coordinate value and the status, to store functional data assigned to the menu in a memory means 23. A conversion status determining means 24 determines that the inputted status coincide with the status to be converted stored in the memory means 23, to notify coincidence to a conversion status output means 25. Receiving the notification, the conversion status output means 25 outputs data to be outputted stored in the memory means 23 instead of inputted switch status. (Abstract)

However, the Morita reference is entirely silent with regard to and does not even suggest that pen data from a digitizer is handled by both unmanaged and managed code on a computing

device as is now recited in claims 1, 8, and 14, where the unmanaged code is code native to and executed directly by a processor of the computing device, and the managed code is code executed in a common language run-time environment of a framework operating on the computing device, where the common language run-time environment of the framework executes the managed code independent of a type of the processor of the computing device.

Moreover, the Morita reference is also entirely silent with regard to and does not even suggest that information relating to such pen data is transferred between unmanaged code and managed code by way of a shared memory and a pointer in the manner recited in claims 1, 8, and 14, where pen data is received in a component on the computing device written in unmanaged code, information related to the pen data is transferred to a shared memory on the computing device designated to be shared between unmanaged code and managed code, a component written in unmanaged code transfers a pointer that points to the information in said shared memory to an application on the computing device written at least in part in managed code, and the application written at least in part in managed code retrieves the information from the shared memory by way of the transferred pointer.

Finally, the Morita reference does not at all appreciate or even suggest that by employing the shared memory in the manner recited in claims 1, 8, and 14, throughput may be increased by a factor of 2-5 times as compared with the standard CLR interop scheme.

Thus, Applicants respectfully submit that the Morita reference fails to disclose or suggest all of the above-mentioned subject matter as are now recited in amended independent claims 1, 8, and 14. Therefore, Applicants respectfully submit that the Morita reference cannot be applied to anticipate or even make obvious the subject matter recited in the independent claims as amended or any claims depending therefrom, including claims 2, 4-7, 9, 11-13, 15, 17-20. Moreover, inasmuch as claims 1, 8, and 14 have been shown to be unanticipated and are non-obvious, then so too must all claims depending therefrom including claims 3, 10, and 16 be

**DOCKET NO.:** MSFT-6146/304450.01  
**Application No.:** 10/692,004  
**Office Action Dated:** May 29, 2008

**PATENT**

unanticipated and non-obvious, at least by their dependencies. Accordingly, Applicants respectfully request reconsideration and withdrawal of the Section 102 and 103 rejections.

**DOCKET NO.:** MSFT-6146/304450.01  
**Application No.:** 10/692,004  
**Office Action Dated:** May 29, 2008

**PATENT**

### **CONCLUSION**

In view of the foregoing Amendment and Remarks, Applicants respectfully submit that the present application including claims 1-20 is in condition for allowance and such action is respectfully requested.

Respectfully Submitted,

Date: August 26, 2008

**/Joseph F. Oriti/**  
Joseph F. Oriti  
Registration No. 47,835

Woodcock Washburn LLP  
Cira Centre  
2929 Arch Street, 12th Floor  
Philadelphia, PA 19104-2891  
Telephone: (215) 568-3100  
Facsimile: (215) 568-3439